

BUILDING GENOME EXPRESSION MODELS USING MICROARRAY EXPRESSION DATA AND TEXT

MICHAEL MOLLA

*University of Wisconsin-Madison,
1300 University Ave., Madison, WI 53703*

PETER ANDRAE

Victoria University of Wellington, NZ

JUDE SHAVLIK

University of Wisconsin-Madison,

Microarray expression data is being generated by the gigabyte all over the world with undoubted exponential increases to come. Annotated genomic data is also rapidly pouring into public databases. Our goal is to develop automated ways of combining these two sources of information to produce insight into the operation of cells under various conditions. Our approach is to use machine-learning techniques to identify characteristics of genes that are up-regulated or down-regulated in a particular microarray experiment. We seek models that are both accurate and easy to interpret. This paper explores the effectiveness of two algorithms for this task: PFOIL (a standard machine-learning rule-building algorithm) and GORB (a new rule-building algorithm devised by us). We use a *permutation test* to evaluate the statistical significance of the learned models. The paper reports on experiments using actual *E. coli* microarray data, discussing the strengths and weaknesses of the two algorithms and demonstrating the trade-offs between accuracy and comprehensibility.

1. Introduction

RNA is the medium by which an organism's genes produce specific proteins, which are the building blocks of life. An understanding of how an organism regulates the production of specific RNA sequences is crucial to an understanding of the mechanism by which that organism functions. The *expression level* of a gene is a measure of the amount of RNA being produced by that gene at a particular time. *Microarrays* are a way to quickly and inexpensively measure the expression levels of thousands of genes simultaneously. Microarrays employ fluorescently labeled fragments of RNA that bind to known locations on the microarray's surface. A scanning laser measures the intensity of fluorescence at each point on that surface. The levels of expression of specific RNAs can be inferred from the intensity values measured by the laser. Microarrays are often used to measure expression levels before and after a specific physical event to see which genes changed their expression levels in response to that event. Genes whose expression levels increased are said to be *up-regulated*, while those whose levels decreased are said to be *down-regulated*.

The development of microarrays and their associated large collections of experimental data have led to the need for automated methods that assist in the interpretation of microarray-based biomedical experiments. We present a method for creating partial interpretations of microarray experiments that combine the expression-level data with textual information about individual genes. These interpretations consist of models that characterize the genes whose expression levels were up- (or down-) regulated. The goal of the models is to assist a human scientist in understanding the results of an experiment. Our approach is to use machine learning to create models that are both accurate and comprehensible.

In order to make them comprehensible, our models are expressed in terms of English words from text descriptions of individual genes. We currently get these descriptions from the curated SwissProt protein database (Bairoch & Apweiler, 2000). It contains annotations of proteins; we use the text associated with the protein generated by a gene as the description of that gene. Our models consist of sets of words from these descriptions that characterize the up-regulated or down-regulated genes. Note that we can use the same text descriptions of the genes to generate interpretations of many different microarray experiments—in each experiment, different genes will be up-regulated or down-regulated, even though the text description associated with each gene is the same across all experiments.

The basic task can be described as follows:

- Given: (a) The (numeric) RNA-expression levels of each gene on a gene array under two conditions, *before* and *after* a particular event (e.g., antibiotic treatment), and
(b) For each gene on the microarray, the SwissProt text describing the protein produced by that gene.
- Produce: A model that accurately characterizes the genes that were *up-regulated* or *down-regulated* in response to the event.

In this article, our models are sets of disjunctive IF-THEN rules of the form:

IF $Word_1$ and $Word_2$ appear in the gene's annotation and $Word_3$ and $Word_4$ are not present
THEN this gene is up-regulated.

For shorthand, in the remainder of this paper we will only present the IF part of the rules and we always focus on the up-regulated group (an arbitrary choice). We would list the above rule as:

$Word_1$ and $Word_2$ and NOT $Word_3$ and NOT $Word_4$

Since our rules are disjunctive, if *any* of the rules match a gene's annotation, our model characterizes that gene as up-regulated. If no rule matches, then our model characterizes that gene as down-regulated.

Our work is related to several recent attempts to use machine learning to predict gene-regulation levels (e.g., Brown et al, 2000, DuDoit et al., 2000; Xing et al., 2001), but our focus is different in that our goal is not to *predict* gene-regulation levels, but to automatically generate human-readable characterizations of the up- or down-regulated genes to help scientists generate hypotheses to explain experiments.

We investigate herein a new rule-building algorithm of our own design against a standard successful algorithm from the machine-learning literature, evaluating how well each satisfies our desiderata of accuracy and comprehensibility. The standard approach to which we compare is PFOIL, a rule learner based on propositional logic.

In our current set of experiments, we consider a gene up-regulated if its ratio of $\text{RNA}_{\text{after}}$ to $\text{RNA}_{\text{before}}$ (the gene's expression ratio) is greater than 2; if this ratio is less than $\frac{1}{2}$ we consider it down-regulated. As is commonly done, we currently discard as ambiguous all genes whose expression ratio is between $\frac{1}{2}$ and 2. We train the learners only using the data set of up-regulated and down-regulated genes and do not attempt to model the ambiguous genes.

In a previous paper (Molla et al., 2002) we described experiments that pitted two successful, standard, machine-learning algorithms against each other at the same task of interpreting gene chip expression data using text annotating the genes. The two algorithms were PFOIL (Mooney 1995) and Naïve Bayes (Mitchell 1997). In this paper we introduce a new algorithm, GORB. We also propose a new evaluation method for machine learning algorithms. In our previous paper, we used cross-validation, the dominant technique for machine-learning evaluation. In this paper we argue that a statistical method called the *permutation test* is actually a more appropriate metric for characterization tasks and that, though cross-validation is still a good measure for predictive accuracy, the permutation test could be applied to other learning tasks where a model or *characterization* of the data, rather than an accurate predictor, is the desired output.

We use this method to measure our first desired property, *accuracy*. We record the accuracy of the model in classifying the examples in the entire data set. This is in contrast to our previous work, which recorded the accuracy of models on a held-out test set to ensure that the accuracy measurement was statistically valid. Instead of using a test set. In this paper, we repeatedly randomly permute the labels of the examples, train the learner and then record the accuracy of the resulting model each time. We only consider the model from the real data to be significant if the accuracy of the model is significantly better than the accuracy of the models from the permuted data sets. Section 2.3 further explains the permutation test.

Our second desired property is human *comprehensibility*. As mentioned before, comprehensibility is the reason we express the rules in terms of English words. The actual comprehensibility of a particular model, however, is very difficult to measure. We use a crude approximation by counting the number of distinct SwissProt words appearing in a given model.

The next section presents the machine-learning algorithms we investigate in our experiments. Section 3 further explains our experimental methodology and Section 4 presents and discusses experimental results obtained using data from the Blattner *E. coli* Laboratory at the University of Wisconsin. Section 5 describes some related work. The final section describes some of our planned follow-up research and

summarizes the lessons learned so far in our work to create a tool that uses English-text protein annotations to assist in the interpretation of microarray experiments.

2. Algorithm Descriptions

This section describes the two algorithms—PFOIL (Mooney, 1995)—and GORB (General-purpose One-step-look-ahead Rule Builder). Both algorithms take as input a collection of training instances (in our case, genes), labeled as belonging to one of two classes (which we will call *up* and *down*), and described by a vector of Boolean-valued features. Each feature corresponds to a word being present or absent from the text description of the gene. Both algorithms produce a model that can be used to categorize a gene on the basis of its feature values (i.e., the words describing it).

2.1 PFoil

PFOIL (Mooney, 1995) is a propositional version of FOIL (Quinlan 1990), a rule-building algorithm that incrementally builds rules that characterize the instances of a class in a data set. FOIL builds rules for a first-order logic language, so that the rules are conjunctions of features (in this case English words, or their negation) that may contain logical variables (and may even be recursive) and must be interpreted by a first-order reasoning engine such as Prolog. PFOIL uses a simpler propositional language, and builds rules that are conjunctions of features. PFOIL rules can be interpreted straightforwardly—a rule covers an instance if each feature in the rule is true of the instance. In our domain, a rule specifies words that must or must not be present in a gene’s annotation.

PFoil builds a set of rules by constructing one rule at a time. It constructs each rule by adding one feature (or its negation) at a time to the current rule. At each step, it chooses the feature that maximizes the performance of the rule according to the FoilGain measure. It stops adding to a rule when either the rule covers only positive instances, or none of the remaining features have a positive FoilGain. When a rule is complete, the algorithm removes all of the positive instances covered by that rule from the data set, and then builds a new rule if there are any positive examples not yet covered by at least one learned rule.

FoilGain is a measure of the improvement that would be obtained by adding a new feature to a rule. It is a trade-off between the coverage of the new rule—the number of positive instances of the class that are covered by the rule—and the increase in precision of the rule—the fraction of the instances covered by the rule that are positive:

$$\text{FoilGain}(\text{rule}, f) = p \cdot [\log(p/(p+n)) - \log(P/(P+N))]$$

where P and N are the number of positive and negative instances covered by rule, and p and n are the number of positive and negative instances that are covered when feature f is added to rule.

As originally described by Mooney (1995), PFOIL does not prune its rule-set. Because PFOIL keeps constructing rules until it has covered all the positive instances, a data set with noise is likely to result in a large set of rules, many of which may be very specific to particular instances.

To address this problem, we have extended PFOIL to include a rule-pruning stage, along the lines of the pruning in FOIL. In the pruning stage, the algorithm repeatedly removes a single feature from one of the rules, choosing the feature whose removal results in the highest accuracy of the remaining rule set. When all the features are removed from a rule, the rule is removed from the rule set. Rather than halting the pruning when the accuracy peaks, in our experiments we continue the pruning until the rule set is empty in order to explore the trade-off between comprehensibility and accuracy.

2.2 GORB

The GORB algorithm (Table 1.) is very similar to that of PFOIL in terms of its input and output. It searches the identical hypothesis space of possible rules, but differs in how it searches this space.

Like PFOIL, GORB explores the hypothesis space by adding one feature at a time to an ever-expanding disjunction of conjunctive rules. One difference, however, is that instead of building the rules sequentially, one rule at a time, GORB considers adding a feature to any existing rule or starting a new rule. This is illustrated in Table 2 with each letter representing one feature. At each step, the current rule set is illustrated.

The other difference is that, instead of using an information-gain-based heuristic to decide which feature to add, GORB computes the accuracy that will result from the addition of this feature. Though time-consuming, this method directly seeks to improve accuracy, which is our desired property.

As with our version of PFOIL, we have included a pruning phase. It works identically to our PFOIL pruning stage, repeatedly removing a single feature from one of the rules, choosing the feature whose removal results in the highest accuracy of the remaining rule set. It is worth noting that this phase, in both PFOIL and GORB, is essentially GORB's hypothesis space search in reverse. Instead of searching the space of possible features for the one whose *addition* results in the best accuracy, the pruning algorithm searches the space of features included in the model for the one whose *removal* results in the best accuracy.

Table 1. Rule construction with GORB

```

Start with rule-set = {}, prevAccuracy = 0, curAccuracy = 1
While curAccuracy > prevAccuracy
    prevAccuracy = curAccuracy
    curAccuracy = 0
    For each feature
        Measure the accuracy of the rule-set on the data with a new rule added consisting
        only of the current feature: eg. "If feature then up-regulated"
        If this accuracy > curAccuracy, set curAccuracy to this value
        For each rule in rule-set
            Measure the accuracy of the rule set on the data with the current
            feature added to the current rule, eg:
                If the rule was:    "If P and Q then up-regulated",
                try the rule:      "If P and Q and feature then up-regulated"
            If this accuracy > curAccuracy, Set curAccuracy to this value
            Repeat the above process for the negation of the current feature
        If curAccuracy > prevAccuracy
            Add the feature that generated curAccuracy to the rule in rule-set (either new or
            existing) where curAccuracy
            was measured
    Return rule-set
    
```

Table 2. Hypothesis space search: PFOIL vs. GORB

	<u>Step 1</u>	<u>Step 2</u>	<u>Step 3</u>	<u>Step 4</u>	<u>Step 5</u>	<u>Step 6</u>
PFOIL	Rule1:A	Rule1:AB	Rule1:AB Rule2:C	Rule1:AB Rule2:CD	Rule1:AB Rule2:CDE	Rule1:AB Rule2:CDE Rule3:F
GORB	Rule1:A	Rule1:A Rule2:B	Rule1:AC Rule2:B	Rule1:AC Rule2:B Rule3:C	Rule1:ACE Rule2:B Rule3:C	Rule1:ACE Rule2:BF Rule3:C

2.3 The Permutation Test

A property of both PFOIL and GORB is that they are guaranteed to find a model for any data set¹, regardless of whether there is any relationship between the descriptions and the labels, because the space of disjunctions of conjunctive rules is large enough to describe any set of instances with any labels. In the worst case, the algorithms could generate a distinct rule for each instance. In most data sets,

¹ Except for the pathological case where two genes have the identical annotation, but one is up-regulated and the other is down-regulated.

especially if the instances have many features, the algorithms will be able to find more compact models that exploit possibly random associations between instances and labels. Therefore, the fact that a model is produced by one of the algorithms is not evidence that the model represents a meaningful relationship between the descriptions and the labels. To show that a model is significant, one must show that the algorithm is very unlikely to have produced a model of the same quality as a result of random associations.

A standard way of showing the significance of a model in a classification or prediction task is to test its accuracy on a held-out test set, since a model that is merely the result of chance associations in the training set will perform poorly on the test set. This approach is very appropriate for prediction tasks, since the accuracy of the model on the test set is also a good estimate of the accuracy on future instances which is important in a prediction task. In a characterization task, there are no future instances to predict, and the training data is the complete data set. Therefore measuring accuracy on a test set is not a useful measure. Furthermore, the limited size of the data set means that holding any data out of the training set will likely reduce the quality of any models that are learned. A *permutation test* is a better way of measuring the significance for a characterization task since it does not require holding out any data and is unrelated to prediction.

A permutation test (Good 2001) is a statistical test to determine significance by comparing the results of an algorithm on a real data set to the results of the algorithm on permutations of the real data set in which the meaningful relationships have been lost. If the model from the real data set is no better than the models from the permuted data sets, then it is not considered significant. If the model from the real data set is much better than models from the permuted data sets, then the model must be taking advantage of semantically meaningful relationships in the data set and is considered significant.

In our application, the quality of a model is its accuracy on the data set given the size of the model. Our permutation test compares the accuracy of a model on the real data set to the accuracy of the models of the same size produced on data sets obtained by randomly permuting the labels (ie, whether the gene is *up-* or *down-regulated*) of the real data set. The model from the real data set is considered significant if its accuracy is clearly higher than the accuracies of all the models from the permuted data sets.. It is important in any permutation test that enough permutations are considered to obtain a statistically valid result. The number of permutations required depends on the data. We have found that 30 permutations are adequate for our particular application².

² One can also obtain a *measure* of the significance by determining the probability that a random, permuted data set will result in a model as good as the real model. Obtaining an accurate measure would require many more permutations than we have used, and we have not yet pursued this approach.

3. Experimental Methodology

The data we are using are from microarray experiments performed by the Blattner *E. coli* Sequencing Laboratory at the University of Wisconsin. In our current computational experiments, we used our methods on 43 different experiments that measure expression of approximately forty-two hundred genes in *E. coli* under various conditions. These conditions include heat and cold shock and various antibiotics for various periods of time. In order to measure the change in expression due to each condition, we compare these expression levels to mean of those measured in six replicate microarrays measured under standard conditions³. By this definition, each experiment includes, on average, 717 up-regulated genes, 352 down-regulated genes and 3221 unregulated genes.

To construct the text description for the genes, we use all words of all the text fields in the SwissProt database. These include the comment (CC) fields (with the exception of the database (CDB) and mass spectrometry (CMS) topics), the description (DE) field, the Organism Classification (OC) field, the keyword (KW) field, and the reference title (RT) fields (with the exception of titles containing: *The Complete Sequence of the E. coli K12 Genome*).

We implemented the algorithms ourselves in the C programming language.

4. Experimental Results and Discussion

Table 3 shows the rulesets of PFOIL and GORB on a typical run on one of our antibiotic testbeds after the all features have been added and the rule sets have been pruned to 10 features each. As it shows, the format of the models is similar, but the content is not always so. In this case, the models generated by the two algorithms share only one word, which might seem surprising given that they are characterizing the same experiment. That word, “biosynthesis,” likely reflects an underrepresentation of basic biosynthetic genes among those up-regulated. Ec2.7.7” and “symport” are interesting rules which each comprise a rule predicting up-regulation in the GORB model. “Ec2.7.7”, is a coded designation by the *Enzyme Commission*, an organization that classifies enzymes by their function. “Ec2.7.7” is the code for nucleotidyltransferases, or enzymes that help to move nucleotides. “Symport” is mechanism of molecular transport. Their up-regulation may provide a hint at what types of processes are involved in the *E. coli* antibiotic shock response.

Figures 1 and 2 show the results of the permutation tests on the two algorithms on one of the experimental data sets and the results of each step of pruning on the two algorithms on the same data set. All data points are plotted. The baseline

³ See <http://www.genome.wisc.edu> for more details about these experiments and the data itself.

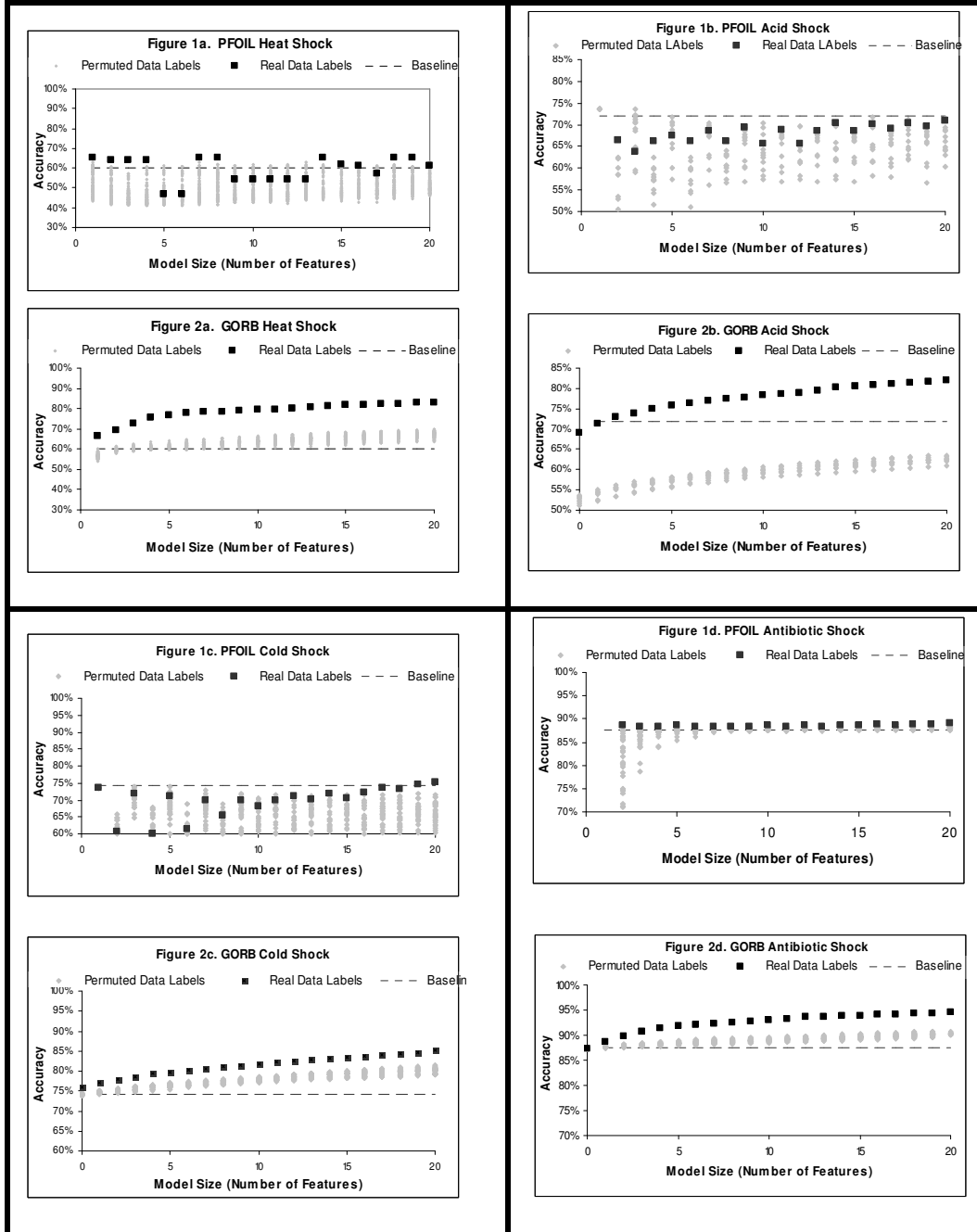
Table 3. Sample Small Rule-Sets Built From the Experimental Cold Shock Data	
<u>PFOIL</u>	<u>GORB</u>
Disjunctive Rules for <i>up</i>	Disjunctive Rules for <i>up</i>
Rule 1: <i>an</i> Rule 2: NOT <i>transport</i> AND <i>membrane</i> Rule 3: <i>hypothetical</i> AND <i>in</i> Rule 3: <i>hypothetical</i> AND <i>gene</i> Rule 4: <i>oxidoreductase</i> AND NOT <i>ec</i>	Rule 1: <i>ribosomal</i> Rule 2: <i>factor</i> NOT <i>control</i> AND NOT <i>activation</i> Rule 3: <i>similarity</i> AND NOT <i>structural</i> Rule 4: <i>mutation</i> AND NOT <i>at</i> Rule 5: <i>sos</i> Rule 6: <i>RNAbinding</i> Rule 7: <i>similar</i>

accuracy for this experiment is 60%. This is the accuracy that would be achieved if the learner always chose the most frequent category (i.e., up-regulated).

On both the real data and the permuted data, PFOIL tends to make much larger models. Though the accuracy of PFOIL's complete model on real data is comparable to the accuracy of GORB's, when models of similar size are compared, GORB is substantially more accurate. For example, GORB's complete rule set contains only 31 words and is 87% accurate at describing the data set. By the time PFOIL's rule set is pruned down to 31 rules, its accuracy has dropped to 68%.

Also striking is the fact that, though both algorithms perform much better on the real data than on permuted data, GORB's margin is much wider and remains so well into the range of comprehensibility (around 10 words). This is of crucial importance because, as explained earlier, this means that GORB is making models that rely on real patterns in the data since, when those patterns are removed, GORB performs much worse.

Figure 3 shows the average amount by which the performance on learning the real data beats the performance on permuted data for all rule sizes between 1 and 20. The values in Figure 3 represent the gaps between the *Permuted Data Labels* line and the *Real Data Labels* line in Figures 1 and 2 averaged across the different 43 experimental conditions. These numbers are important because they measure the extent to which the learner's performance depends on real regularities in the data as opposed to absolute accuracy which can come from other sources such as overfitting. GORB seems to do much better than PFOIL in this respect. In fact, for models whose size is smaller than 5, PFOIL tends to perform better on the permuted data than on the real data. We do not presently have an explanation for this behavior except that it has occurred by random chance.



Discussion

The structure of the GORB rule set in Table 3 is typical of those generated by GORB. That is, the first rule starts with a high frequency word (in this case “protein”) that would give high coverage but low accuracy. The rest of the rule contains further features, usually negations of less frequently occurring words that specialize the rule to improve accuracy. The other rules are more specialized rules consisting of only one or two words that each cover a much smaller number of the instances that are not covered by the first rule. The features in these rules are generally low frequency words and the rules have high accuracy.

During its pruning process, PFOIL can also generate rules consisting of just a few features. However, these rules typically have low accuracy, because they are created by pruning out the “specializing” features from rules. A consequence is that the PFOIL pruning process results in a very bumpy accuracy curve, with large dips in accuracy when a rule is pruned to a single feature, followed by a rise in accuracy when the rule is removed. On the other hand, the GORB pruning curves are much smoother, since GORB constructs the small rules in a different way. We believe this smoothness is desirable since it makes a smoother tradeoff between accuracy and comprehensibility, a practical benefit of a tool based on this approach.

Though the GORB rule sets are more accurate and more significant than the PFOIL rule sets across the whole range of rule set sizes, the difference in the very heavily pruned models is most striking. GORB appears to be much better at characterizing the data when only allowed to use a few words in its models. This suggests that it is doing a better job than PFOIL at identifying the *real* regularities in the data.

Though more accurate, the GORB strategy does suffer from having a large number of rules that are only relevant in the context of the greater rule set. This is because they only specify a single infrequently occurring word that may or may not be descriptive on its own; that is on the whole set of genes, not just those not covered by other rules. One remedy for this may be an algorithm that is biased toward rules of similar length, or individual rules that have high precision even if they have only moderate coverage

By our measure, the comprehensibility of both methods is good. Short models tend to be almost as good as longer ones. It is, however, also clear that, as mentioned earlier, length is a crude measure. Though we probably are not ready for it yet, attainment of the final answer on comprehensibility will probably have to involve human testers.

5. Related Work

A great deal of research has been done in text mining, much of which involves biomedical tasks. Hearst's LINDI system (1999) searches medical literature for text relating to a particular subject or problem and tries to make logical connections to form a hypothesis. One of the benefits of our approach is that researchers do not need to know what they are looking for in advance. Given expression data and textual descriptions of the genes represented in the expression data, this system makes an automated "first pass" at discovering what is interesting. The PubGene tool (Jennssen et al., 2001) also interprets gene-expression data based on textual data. One big difference is that PubGene compiles clusters of text ahead of time and tries to match the expression data to an already-made cluster. Our tool is designed to allow the expression data itself to define its models. Masys et al. (2001) also use text associated with genes to explain experiments. However, they cluster expression values across experiments and then use the text to explain the clusters, whereas we use the text directly during the learning process and can explain single experiments.

6. Future Directions and Conclusions

We have presented an approach for aiding in the interpretation of microarray experiments that is based on machine learning and uses SwissProt text as its representation of the microarray's genes. We argue that two important properties for such a computational tool are that it should produce models that are (1) accurate and (2) readable. We empirically compared an algorithm of our own design to a widely successful algorithm—PFOIL—on an *E. coli* microarray experiment, evaluating these two approaches with respect to our desiderata. We also evaluated the significance of the models using a permutation test. We have shown that, though both algorithms are able to find characterizations of the data with reasonable accuracy, our new algorithm does a better job on this data set. Also, our new algorithm produces models that have greater significance than the PFOIL models.

One current limitation to this approach is that it relies only on text, though the sequence data are almost always also available. We plan to explore methods that make use of both the sequence data and the text annotating the genes. Another enhancement would be to increase the textual data we use. Abstracts from appropriate journal articles would be a logical next step. Also, we could consider sequences of words or common phrases instead of only individual words.

Another criterion we consider important is *stability*. Learned models should not change very much if the experiment were repeated under slightly different conditions. We plan to apply a method that takes advantage of replicate data. We previously devised a method for measuring stability without it (Molla et al., 2002).

Another future direction is to move beyond our method of simply using as a two-fold change in expression as our definition of up- and down-regulation. We

intend to apply the method for gauging fold-change described in (Newton et al.,2001) and redo the experiments presented here.

Acknowledgments

We would like to thank Jeremy Glasner and Fred Blattner and the E. Coli lab of the University of Wisconsin-Madison for providing the data and helping us to understand it. This research was funded by grants NLM 1 R01 LM07050-01, NIH 2 P30 CA14520-29, and NIH 5 T32 GM08349.

References

- Bairoch A. & Apweiler R. (2000). The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucl. Acids Res.* 28:45-48.
- Brown, W., Lin, D., Cristianini, D., Sugnet, C., Furey, T., Ares, M. & Haussler, D.(2000).Knowledge-based analysis of microarray gene expression data using support vector machines. *Proc. Natl. Acad. of Science*, 97(1):262-267.
- Dudoit, S., Yang, Y., Callow, M. & Speed, T. (2000). *Statistical Methods for Identifying Differentially Expressed Genes in Replicated cDNA Microarray Experiments*. UC-Berkeley Statistics Dept. TR #578.
- Good, P. (2001), *Resampling Methods A Practical Guide to Data Analysis, 2nd edition*, Birkhduser Boston.
- Hearst, M.(1999).Untangling text data mining.*Proc.37th Annual Meetg. Assoc.for Com Ling.*
- Jenssen, T-K., Lægreid, A., Komorowski, J., & Hovig, E. (2001). A literature network of human genes for high-throughput gene-expression analysis. *Nature Genetics* 28:21-28.
- Manning, C.D., Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press.
- Masys, D. et al. (2001).Use of keyword hierarchies to interpret gene expression patterns. *Bioinformatics*,17:319-326.
- Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.
- Mladeni'c, D. & Grobelnik, M. (1999). *Feature selection for unbalanced class distribution and Naive Bayes* , Proceedings of the 16th International Conference on Machine Learning ICML-99, Morgan Kaufmann Publishers, San Francisco, CA. pp. 258-267.
- Molla, M., Andraea, P., Glasner, J., Blattner, F., & Shavlik, J. (2002). Interpreting Microarray Expression Data Using Text Annotating the Genes *Proceedings of the 4th Conf. on Computational Biology and Genome Informatics*.
- Mooney, R.J. (1995).Encouraging Experimental Results on Learning CNF. *Machine Learning* 19,1:79-92.
- Newton, M.A., Kendziorski, Richmond, C.S., Blattner, F.R., and Tsui, K.W. (2001).On Differential Variability of Expression Ratios: Improving Statistical Inference About Gene Expression Changes From Microarray Data. *Journal of Computational Biology* 8:37-52
- Porter, M. (1980). An algorithm for suffix stripping. *Program*, 14:130-137.
- Quinlan, J. R. (1990). Learning logical descriptions from relations. *Machine Learning*, 5:239-266.
- Xing, E, Jordan, M, & Karp, R. (2001). Feature selection for high-dimensional genomic microarray data. *Proceedings of the International Conference On Machine Learning*, pp.601-608, Morgan Kaufmann.